Interaktion

INTERAKTION: Users Group for the Interak Computer.

Newsletter Number 1 December 1982

C O N T E N T S

Dear Member, at last I have been able to put finger to keyboard in order to get to you the first "Interaktion" news letter. This first attempt will I hope stimulate constructive comments on both content and format. I hope you all find something to interest you here and if not let me know.

PV

Byte Bag:

This is a useful sub routine that has helped me out in the past. It compares the contents of the HL register pair with that of DE. The results are found by examining the condition bits. The routine is of particular interest as neither HL nor DE are altered. The answer is found as follows:-

```
OR A          B7
SBC HL,DE     ED 52
ADD HL,DE     19
RET           C9
```

HL=DE Z FLAG SET HL>=DE CARRY
FLAG RESET HL<DE CARRY FLAG SET
PV

[Do you have any clever little programs?
If you have send them in]

HEX DUMP:

This machine code program will give ZYMON 2 one extra facility. On entering the command letter D followed by a start and stop address the program will dump the Hex code found between the two addresses. PV

```
8000  3E 44 32 FF 07 3E 24 32 00 08 3E 44 32 01 08 3E
8010  21 32 02 08 3E 80 32 03 08 3E FF 32 04 08 C3 66
8020  00 3A DE 0F 6F 3A DF 0F 67 22 3C 80 3A E0 0F 6F
8030  3A E1 0F 67 22 3E 80 C3 40 80 38 39 3C 80 C2 80
8040  26 DD CD B8 80 26 0A CD B8 80 ED 5B 3C 80 21 3A
8050  80 4A CD 18 06 3A 3A 80 67 CD B8 80 3A 3B 80 67
8060  CD B8 80 4B 21 3A 80 CD 18 06 3A 3A 80 67 CD B8
8070  80 3A 3B 80 67 CD B8 80 26 20 CD B8 80 26 20 CD
8080  B8 80 06 10 2A 3C 80 4E 21 3A 80 CD 18 06 3A 3A
8090  80 67 CD B8 80 3A 3B 80 67 CD B8 80 26 20 CD B8
80A0  80 ED 5B 3C 80 2A 3A 80 CD BB 06 CA 74 05 13 ED
80B0  53 3C 80 10 CF C3 40 80 DB 06 CB 7F 28 FA 76 D3
80C0  D7 C9 FF
```

[Have you got any programs to send for the HEX DUMP spot?
Use this program to dump them and send them to me]

BITS OF BASIC.[1]

Strings and Poke.

The strings on ZYBASIC V2.03 are held in store, from 8100 hex upwards. 26 strings exist, each starting on a 256 byte boundary.

A$ 8100	B$ 8200	C$ 8300	D$ 8400
E$ 8500	F$ 8600	G$ 8700	H$ 8800
I$ 8900	J$ 8A00	K$ 8B00	L$ 8C00
M$ 8000	N$ 8E00	O$ 8F00	P$ 9000
Q$ 9100	R$ 9200	S$ 9300	T$ 9400
U$ 9500	V$ 9600	W$ 9700	X$ 9800
Y$ 9900	Z$ 9A00		

To see a string try the following:

```
10 A$="AAAA"
20 Z$="ZZZZ"
RUN
BYE
T 8100
T 9A00
```

and both strings are now on the screen as thay are held in real store. An important fact about the strings is that thay are terminated by an FF character. To see that try:-

```
10 A$="AAAAAA"
20 A$="ZZ"
RUN
BYE
T 8100
```

and the FF code is in the third position. This tells ZYBASIC that A$ is only two characters in length. The above facts mean we can do to strings what very few Basics will allow. Poke to them! The only rule to follow is to poke an FF at the end of the string. A use for all of this is in fast graphic displays. So:-

a string is 256 (-1) characters long, take A$ as an example,8100-81FF. Given that 81FF must be set to FF than we have 255 characters to work with. A VDU line is 32 characters long, and so a string covers 7.6 lines on the screen. If we ignore the 0.6, then 7 screen lines can be written with one PRINT line! ie a string could hold a monster for a graphics game, or two strings could hold the 14 line short range scan display for the game of STARTREK. Updates would be POKEO into the string, and display would be

```
PAGE:PRINT $:PRINT $.
```

Another use is for USR routines, hold them in a OATA statement, read them into the string and USR the string address to execute your code.

To start you off, POKEing into strings, here is a utility to null them.

```
 998 ! Null String S
 999 ! entry S has string address
1000 FOR Z9=0 TO 223      covering 7 lines
1010 POKE S+Z9,0          null a character
1020 NEXT Z9              null 224 characters
1030 POKE S+224,#FF       plant the terminator
1040 RETURN               exit string is nulled
```

Enter with the string address in S, ie take A$ at 8100

```
  10 S=#8100              S points to A$
  20 GOSUB 1000           clear S string,A$
  30 PRINT A$             print th result
  40 STOP                 end
```

For fun,change line 1010 to

```
1010 POKE S+Z9,d
```

and add

```
  15 INPUT 0
  35 GOTO 10
```

Now RUN and give 6 in response to the data request.

Finally try this program which shows how 3 PRINT $ statments can cover the major display area on the screen:-

```
  10 CLS:00FF:PAGE:LINE1
  20 S=#8100
  30 FOR 0=0 TO #7E
  40 GOSUB 1000
  50 P.A$:LINE 8
  60 P.A$:LINE 15
  70 P.A$:LINE 1
  80 NEXT 0
  90 GOTO 20
1000 FOR Z=0 TO 223
1010 POKE Z9+S,D
1020 NEXT Z9
1030 POKE S+224,#FF
1040 RETURN
```

Please note: the character codes 8 and hex 7F are backspace and so produce a funny response when printed as a string. Also code 12 is CTRL L which is clear screen, so you get 672 clear screens! And finally 13 is CR, so you 672 carriage returns!

With those restrictions in mind I hope you get some fun out of POKEing to a string.

BYE
B.E.

CODE BREAKER				ASCII			
ASCII	LOC.	UPC.	CON.	ASCII	LOC.	UPC.	CON.
0	30	00	00				
1-!	31	21	00	u	75	55	15
2-"	32	22	00	v	76	56	16
3-#	33	23	00	w	77	57	17
4-$	34	24	00	x	78	58	18
5-%	35	25	00	y	79	59	19
6-&	36	26	00	z	7A	5A	1A
7-/	37	27	00	--=	3D	2D	00
8-(	38	28	00	^-	7E	5E	1E
9-)	39	29	00	LF	0A	0A	0A
a	61	41	01	BS	08	08	08
b	62	42	02	ESC	1B	1B	1B
c	63	43	03	TAB	09	09	09
d	64	44	04	,-@	60	40	00
e	65	45	05	{-[	7B	5B	1B
f	66	46	06	RUB	7F	7F	7F
g	67	47	07	;-+	3B	2B	00
h	68	48	08	:-*	3A	2A	00
i	69	49	09	}-]	5D	7D	1D
j	6A	4A	0A	RET	0D	0D	0D
k	6B	4B	0B	CUD	0A	0A	0A
l	6C	4C	0C	CUU	0B	0B	0B
m	6D	4D	0D	CUL	08	08	08
n	6E	4E	0E	CUR	09	09	09
o	6F	4F	0F	<-,	2C	3C	00
p	70	50	10	>-.	2E	3E	00
q	71	51	11	?-/	2F	3F	00
r	72	52	12	CLE	18	18	18
s	73	53	13	HOM	0C	0C	0C
t	74	54	14	SPC	20	20	20

(LOC= lower case UPC= upper case)

NOTE some keyboards may not conform to the above layout although the code is standard.

I have been sorting through some old programs to see if there was something interesting to use as the last article in our news letter and by coincidence I came across the following, which uses to advantage the direct addressing of ZYBASIC'S strings as described in the BITS OF BASIC article.

One of the problems when taking strings from the key-board is that the operator may input upper or lower case characters. If a search is necessary then account of this must be taken.

This simple program modifies A$ to be the lower case equivalent of the input. Slight modifications to lines 1040 and 1050 will ensure A$ is always upper case.

```
1000 REM ALL LOWER CASE
1010 A=#80FF
1020 FOR L= 1 TO LEN(A$)
1030 O=PEEK(A+L)
1040 IF O>96 GOTO 1070
1050 O=O+32
1060 POKE A+L,O
1070 NEXT L
1080 RETURN
```

Finally here is a use for my subroutine.

Input an English number and out pops French!
Input a French number and out pops the English!

```
   1 CLS
  10 A$=INPUT$
  12 GOSUB 1000
  15 RESTORE
  20 FOR L=1 TO 30
  30 READ B$
  40 IF A$=B$ GOTO 70
  50 NEXT L
  60 GOTO 10
  70 READ B$:PRINT B$
  80 PRINT A$
  90 GOTO 10
 999 STOP
1000 REM ALL LOWER CASE
1010 A=#80FF
1020 FOR L= 1 TO LEN(A$)
1030 D=PEEK(A+L)
1040 IF D>96 GOTO 1070
1045 IF O<64 GOTO 1070
1050 D=O+32
1060 POKE A+L,O
1070 NEXT L
1080 RETURN
3000 DATA "one","une",one","two","deux","two","three","tr
ois","three","four","quatre","four","five","cinq","five","six","
six","six","seven","sept","seven","eight","huit","eight","nine",
"neuf","nine","ten","dix","ten"
```

To increase the range add data lines, and modify line 20 to the total number of entries in the data files. Note that three entries are needed per word to carry out two way translation. The possibilities for this program are limitless: German, Polish, Norwegian, Swedish, telephone numbers etc. . .

. . . A Word From Our Sponsor. . .

The Greenbank Connection.

Thank you thank you for inviting me into your hearts and homes, by asking me to make some small contribution to the Interaktion Newsletter (and believe me it will be small). Gentlemen, (I understand we have no Lady Interaktions?), I am at your disposal. Do please let Peter (the head Interakter) know what you would like to see in this column, and I shall do my best to oblige. Do you want cosy little fire-side chats (e.g. What to do when you hit return, should you have typed RUN first, and if you forgot would it be a "hit"-and-"RUN" accident?) Or, would you like hard-core circuit diagrams, design discussions, beginner's notes; ask and it shall be given. I can write this guff by the mile, absolutely effortlessly - the only effort I find is having to read it afterwards!

To get us started, see how you like this; is this the sort of thing that the rank and file membership at grass roots level earnestly desire, or shall we enter into meaningful dialogue inter alia, or shall we keep politics out of it?

Mode 2 Interrupts on the Interak 1 System

"I don't mind the odd interruption, but I won't be able to get anything done if you're going to interrupt me all the time." (Z80A-CPU circa 1979).

There's a lot of sense in this, and this is one of the reasons (there are others) why the Interak 1 system as it stands does not use interrupts. Unlike the little old lady who saved her lace table-cloth for best, and found it was all in holes when she finally came to use it, interrupts should be saved for the time you really need them. When you want interrupts, you'll know about it, and the original designers of the system we now know as Interak 1, were quite sensible in leaving them out, rather than dragging them in just because they were there.

The main need for interrupts is felt when it is desired to use the extremely attractive Z80A support chips, e.g. PIO (Peripheral Input Output), CTC (Counter Timer Circuit), and so on. It is almost mandatory to use "Mode 2" interrupts when using these chips, and the resulting speed of response, and magnificent performance this gives, makes you glad that together we have chosen a system which has a Z80A-CPU.

The Mode 2 interrupt gains its power from the fact that the interrupting device can supply data to form the address of the appropriate service routine in the software, easily and effortlessly, with no need for the CPU to waste any time. In effect, when a conventional CPU (i.e. not the superior Z80A which we all use chaps) receives an interrupt, it has to go wandering around for ages, asking each interrupting device "did you interrupt me?", and once it knows, it has to sort them into

priorities, so as to do the most important ones first, all very tedious and time wasting. On the other hand the Z80A support chips are clever enough to do much of this work themselves (after all the interrupting device itself knows better than anyone that it was the source of the interrupt).

So as not to monopolise this newsletter any more I shall not speak further about why you should want Mode 2 Interrupts, but suffice it to say: they are a GOOD THING.

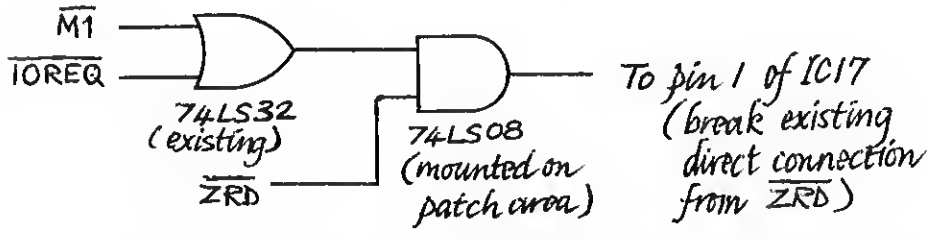
In a manner of speaking, it is as if the interrupting devices, (which are in the I/O space) were actually supplying part of an op-code to the CPU, and in fact this is pretty well what happens. On receipt of an interrupt request from some device (we know not what), the Z80A generates a special "interrupt acknowledge" cycle. This special cycle takes the "$\overline{M1}$" line low (i.e. the line which normally goes low when an op-code is being fetched from the memory comprising the program store), but instead of the "$\overline{MREQ}$" line going low to access the memory space, the "$\overline{IOREQ}$" line goes low, and the interrupting device places its "interrupt vector" on the data bus, i.e. the special cycle acts as though the CPU was fetching an op-code from the I/O space.

It is a fairly simple matter to modify the standard "Kemitron" card MZB-3 so that Mode 2 interrupt vectors can be accepted:

The data bus tranceiver on the MZB-3 card has its pin 1 (direction control) connected directly to the signal $\overline{ZRD}$ (see MZB-3 Manual for details). This alone will not control the transceiver correctly for Mode 2 interrupt acknowledgement, since this does not generate $\overline{ZRD}$. An extra 74LS08 gate can be added, and an unused portion of a 74LS32 gate already on the card be used in the circuit below (figure 1). It has the required action because the bus tranceiver pin 1 control line now goes low when either $\overline{ZRD}$ goes low (as before) or when both $\overline{M1}$ and $\overline{IOREQ}$ are low (i.e. interrupt acknowledgement). It is possible, a customer tells us, to produce the same logic without adding the extra 74LS08, but this requires more track cutting and a certain amount of gate transplantation, and is thus messier to perform and describe.

D.M.P.

Figure 1



(Modification to MZB-3 Card to Permit Use of Mode 2 Interrupts.)

MEMBERS' LETTERS.

Brian Hunter
13 Holefarm Rd
Renfrewshire
Tel Greenok 2490

Would like to hear from any member with second hand, built boards for sale.

Alan Jenkins
55 Ash Grove
Craigshill,Livingston
West Lothian

Is converting an 8 track tape to floppy-type system. Anyone interested in helping?

Michel Orton
73 Fairacres Rd
Oxford
Tel 0865 49154

Is any one else trying to adapt P. L. Woods "cheap line printer" to ISBUS?

FOR SALE

RCA touch sensitive Key-Board £15. Phone Stafford 760 250

Well that's all for now, please help by sending in your letters, programs, wants, sales, problems and comments.

PV